REPORT

안드로이드 악성 앱 분석 자동화 프로그램 구현

교 과 목	SW보안개론
분 반	1
담 당 교 수	조성제 교수님
학 과	소프트웨어학과
학 번 이 름	32141847 박영민
	32141868 박유현
	32143180 이명재
	32144697 최광진
제 출 일	19.05.13

목 차

1. 개요

1.1 정상/악성 앱의 동작 방식, 정상/악성 앱을 판단하는 기준 (signature)

- 배포해주신 feature_extractor 소스코드를 사용해서 제공받은 앱들의 권한을 확인해본 결과 악성 앱과 그렇지 않은 앱들 사이에 권한의 차이가 있는 것을 발견했다. 따라서 악성 앱에만 존재하는 권한들을 추출하고 데이터베이스를 이용해 (MySQL프로그램 이용) 분석해내어 판단하는 기준으로 삼았습니다.
- 또한 정확한 판단을 위해 Virus Total 사이트에 접속하여 직접 파일을 업로드 하여 정상/악성 앱을 분류할 수 있었습니다.

1.2 악성 앱 분석 자동화 프로그램의 알고리즘 개요 및 설명

- feature_extractor 소스코드가 각 앱의 AndroidManifest.xml에 담겨있는 권한들을 Androguard를 사용하여 검색하는 프로그램이기 때문에 기존의 검색을 하는 기능에 악성 앱에만 존재하는 특징적인 권한들이 존재하는지 확인하는 소스코드를 추가하여 그 권한들이 있다면 악성 앱으로 분류하는 알고리즘을 적용하였습니다. 악성 앱에만 존재하는 권한들 중에 정상 앱에도 존재하는 권한이지만 그에 대한 세부 권한이 존재하는 것 들이 있기 때문에 이점을 유의하여 정확히 구분해야 할 필요가 있으므로 이에 중점을 두어 알고리즘을 만들었습니다.

1.3 구현 환경 및 구현 과정

- ■ feature_extractor를 사용하여 앱이 갖고 있는 권한 분석
- ■ Windows에서 Jadx를 이용하여 디컴파일 및 분석
- ■ Virus Total을 사용하여 의심되는 앱 상세분석
- ■ Windows에서 안드로이드 스튜디오 에뮬레이터 이용하여 실행 확인
- ■ Linux Ubuntu LTS16.04 환경에서 기존의 feature_extractor를 이용하여 새로운 악성 앱 탐지 프로그램을 파이썬 코드로 작성

2. 구현 과정 상세

2.1 기존 프로그램을 통해 얻은 데이터 DB화

MySQL을 이용하여 우리가 원하는 조건에 맞는 permission 찾는 방법을 사용하였습니다. 가장 먼저 한 일은 feature_extractor의 결과파일인 res.csv를 검색하기 용이하도록 file name과 permission으로 분류하여 DB화 한 것입니다.

Field	Type *	Collation	Null *	Key *	Default	Extra	Privileges	Comment *
fileName	varchar(500)	utf8mb4_0900_ai_ci	YES		{null}		select,insert,update,references	
permission	varchar(500)	utf8mb4_0900_ai_ci	YES		{null}		select,insert,update,references	

<테이블 Column>

fileName	permission
./benign_app/com.android.music_2.apk	android.permission.BROADCAST_STICKY
./benign_app/com.android.music_2.apk	android.permission.INTERNET
./benign_app/com.android.music_2.apk	android.permission.SYSTEM_ALERT_WINDOW
./benign_app/com.android.music_2.apk	android.permission.WAKE_LOCK
./benign_app/com.android.music_2.apk	android.permission.WRITE_SETTINGS
./benign_app/com.android.music_2.apk	android.permission.READ_PHONE_STATE
./benign_app/com.android.music_2.apk	android.permission.WRITE_EXTERNAL_STORAGE
./benign_app/com.android2.calculator3_93.apk	android.permission.VIBRATE
./benign_app/com.android2.calculator3_93.apk	android.permission.SYSTEM_ALERT_WINDOW
./benign_app/com.android2.calculator3_93.apk	android.permission.WAKE_LOCK
./benign_app/com.android2.calculator3_93.apk	android.permission.INTERNET
./benign_app/com.android2.calculator3_93.apk	com.android.vending.BILLING
./benign_app/com.android2.calculator3_93.apk	android.permission.GET_ACCOUNTS
./benign_app/com.androidemu.nes_61.apk	android.permission.VIBRATE
./benign_app/com.androidemu.nes_61.apk	android.permission.ACCESS_WIFI_STATE
./benign_app/com.androidemu.nes_61.apk	android.permission.BLUETOOTH
./benign_app/com.androidemu.nes_61.apk	android.permission.INTERNET
./benign_app/com.androidemu.nes_61.apk	android.permission.WRITE_EXTERNAL_STORAGE
./benign_app/com.androidemu.nes_61.apk	android.permission.BLUETOOTH_ADMIN
./benign_app/com.angryburg.uapp_421.apk	android.permission.INTERNET
./benign_app/com.angrydoughnuts.android.alarmclock_15.apk	android.permission.WAKE_LOCK

<테이블에 추가되어 있는 데이터>

2.2 DB화 한 데이터를 이용하여 의미 분석

DB화 이후 악성 앱에만 있는 권한(permission)을 찾을 수 있도록 SQL쿼리문을 작성하였습니다.

```
SELECT
 permission,
 count(1) cnt
FROM
 app_permission
WHERE
 fileName LIKE '%malware%' AND
 permission NOT IN (SELECT
                      DISTINCT permission
                    FROM
                      app_permission
                    WHERE
                      fileName LIKE '%benign%')
GROUP BY
 permission
```

2.2.1 악성 앱에만 있고 정상 앱에는 없는 권한 쿼리 수행결과

Permission	cnt
android.permission.ACCESS_CACHE_FILESYSTEM	1
android.permission.ACCESS_COARSE_UPDATES	2
android.permission.ACCESS_GPS	10
android.permission.ACCESS_LOCATION	8
android.permission.BROADCAST_PACKAGE_REMOVED	9
android.permission.CAMERA	6
android.permission.CHANGE_NETWORK_STATE	9
android.permission.CLEAR_APP_CACHE	1
android.permission.DELETE_CACHE_FILES	2
android.permission.DELETE_PACKAGES	5
android.permission.DEVICE_POWER	2
android.permission.FLASHLIGHT	2
android.permission.GET_PACKAGE_SIZE	1
android.permission.INSTALL_PACKAGES	14
android.permission.MOUNT_UNMOUNT_FILESYSTEMS	21
android.permission.PERMISSION_NAME	2
android.permission.PROCESS_OUTGOING_CALLS	7
android.permission.READ_OWNER_DATA	2
android.permission.REORDER_TASKS	2
android.permission.RESTART_PACKAGES	23
android.permission.WRITE_APN_SETTINGS	20

android.permission.WRITE_OWNER_DATA	2
android.permission.WRITE_SECURE_SETTINGS	1
com.android.launcher.permission.UNINSTALL_SHORTCUT	12
com.android.launcher.permission.WRITE_SETTINGS	11
com.android.vending.CHECK_LICENSE	3
com.htc.launcher.permission.READ_SETTINGS	11
com.lge.launcher.permission.INSTALL_SHORTCUT	11
com.lge.launcher.permission.READ_SETTINGS	11
com.lge.launcher.permission.UNINSTALL_SHORTCUT	11
com.lge.launcher.permission.WRITE_SETTINGS	11
com.motorola.dlauncher.permission.INSTALL_SHORTCUT	5
com.motorola.dlauncher.permission.READ_SETTINGS	5
com.motorola.dlauncher.permission.UNINSTALL_SHORTCUT	5
com.motorola.dlauncher.permission.WRITE_SETTINGS	5
com.motorola.launcher.permission.INSTALL_SHORTCUT	11
com.motorola.launcher.permission.READ_SETTINGS	11
com.motorola.launcher.permission.UNINSTALL_SHORTCUT	11
com.motorola.launcher.permission.WRITE_SETTINGS	11

그 다음 정상 앱과 겹치는 권한이 있을 수 있기에, 악성 앱의 권한 중 악성 앱, 정상 앱 모두에 있는 권한을 찾는 SQL쿼리문을 작성하였습니다.

```
SELECT
 permission,
 count(1) cnt
FROM
 app_permission
WHERE
 fileName LIKE '%malware%' AND
 permission IN (SELECT
                 DISTINCT permission
                FROM
                 app_permission
                WHERE
                 fileName LIKE '%malware%') AND
 permission IN (SELECT
                 DISTINCT permission
                FROM
                 app_permission
                WHERE
                 fileName LIKE '%benign%')
GROUP BY
 permission
```

2.2.2 정상 앱과 악성 앱에 전부 있는 권한 쿼리 수행결과

Permission	cnt
android.permission.RECEIVE_BOOT_COMPLETED	77
android.permission.WAKE_LOCK	46
android.permission.WRITE_EXTERNAL_STORAGE	96
android.permission.ACCESS_FINE_LOCATION	47
android.permission.INTERNET	142
android.permission.DISABLE_KEYGUARD	4
android.permission.VIBRATE	27
android.permission.CALL_PHONE	16
android.permission.ACCESS_NETWORK_STATE	75
android.permission.BLUETOOTH	11
android.permission.ACCESS_COARSE_LOCATION	44
android.permission.SEND_SMS	34
android.permission.READ_LOGS	20
android.permission.READ_PHONE_STATE	94
android.permission.ACCESS_WIFI_STATE	51
android.permission.WRITE_CONTACTS	17
android.permission.CHANGE_WIFI_STATE	20
android.permission.RECORD_AUDIO	7
android.permission.READ_CONTACTS	39
android.permission.GET_TASKS	6
android.permission.SYSTEM_ALERT_WINDOW	4
android.permission.MODIFY_AUDIO_SETTINGS	4
android.permission.GET_ACCOUNTS	15
android.permission.RECEIVE_SMS	31
com.android.browser.permission.WRITE_HISTORY_BOOKMARKS	24
android.permission.BLUETOOTH_ADMIN	8
com.android.launcher.permission.INSTALL_SHORTCUT	25
android.permission.MODIFY_PHONE_STATE	21
android.permission.KILL_BACKGROUND_PROCESSES	7
android.permission.SET_WALLPAPER	15
android.permission.WRITE_SMS	17
android.permission.WRITE_SETTINGS	17
com.android.launcher.permission.READ_SETTINGS	12
com.android.browser.permission.READ_HISTORY_BOOKMARKS	24
android.permission.ACCESS_LOCATION_EXTRA_COMMANDS	2
android.permission.READ_SMS	29

위의 쿼리 결과를 통해 제공받은 정상/악성 앱 데이터셋에 악성 앱으로 분류되어 있지만, 정상권한만을 갖는 앱을 찾기 위한 자바코드를 작성하였습니다.

```java
package main;

import java.util.ArrayList;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import logic.SendSqlDAO;

public class Test {

  public static void main(String[] args) {
    try {
      SendSqlDAO dao = new SendSqlDAO();
      JSONObject permission = dao.sendQuery("select permission from app_permission where filename like '%malware%' and "+
          "permission in (select distinct permission from app_permission where filename like '%benign%') group by permission");
      JSONArray pList = (JSONArray) permission.get("rows");
      JSONObject res = dao.sendQuery("select distinct filename from app_permission where filename like '%malware%'");
      JSONArray malAppList = (JSONArray)res.get("rows");
      ArrayList<String> resAppList = new ArrayList<>();
      for (int i = 0; i < malAppList.size(); i++) {
        String name = (String)((JSONArray)malAppList.get(i)).get(0);
        JSONObject temp = dao.sendQuery("select distinct permission from app_permission where filename='"+name+"'");
        JSONArray tempArr = (JSONArray) temp.get("rows");
        boolean chk = true;
        for (int j = 0; j < tempArr.size(); j++) {
          boolean chk2 = true;
          for (int k = 0; k < pList.size(); k++) {
            String tempPer = (String)((JSONArray)tempArr.get(j)).get(0);
            String pListPer = (String)((JSONArray)pList.get(k)).get(0);
            if (tempPer.equals(pListPer)) {
              chk2 = false;
              break;
            }
          }
          if (chk2)
            chk = false;
        }

        if (chk)
          resAppList.add(name);
      }

      System.out.println("cnt : " + resAppList.size());
      for (String str : resAppList)
        System.out.println(str);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

<Java 소스코드>

2.2.3 정상 권한만을 갖는 악성 앱 검출 코드 수행결과

```
cnt : 30
./malware_app/806aaf7772889ae61d64dd8c40e674fc3db497b7.apk
./malware_app/73ddc408b518826064878dfc0064c4cd4fe512c0.apk
./malware_app/15799032055df9f93440b61547b37e01a7234a0c.apk
./malware_app/0db53aefbd60325df198e9ebb190f45c61ac8923.apk
./malware_app/d64419d569a336f05d0eb799615097870934c42a.apk
./malware_app/005d5f6e94321de473d62706a94fbecf67c9f5f3.apk
./malware_app/0c059ad62b9dbccf8943fe4697f2a6b0cb917548.apk
./malware_app/edbbc205e7033174919ba22b01617fd731960da2.apk
./malware_app/7f11794529652cf3e785e1c51b9ccbf1f00bce4a.apk
./malware_app/d2fdaf2c293adb2977c8d3c7aa7152553021f949.apk
./malware_app/1c0a6b1c5d24cbba9b11020231fffc0840dd7e10.apk
./malware_app/fdf6509b4911485b3f4783a72fde5c27aa9548c7.apk
./malware_app/730fed46dc7f13691906f46111ee5e05aa1b854e.apk
./malware_app/72adcf43e5f945ca9f72064b81dc0062007f0fbf.apk
./malware_app/c6b7ec91f6e237978552a478306fb6e01c9f15e9.apk
./malware_app/a8634a3392690a13146fbfe286e117b3e34bf96e.apk
./malware_app/5a74f1796a4177d2c29be91ce15dea217ec7adc5.apk
./malware_app/38331a6364fba626d99d41de5cf87bcaabe3ab5c.apk
./malware_app/127df235a9181bb4cdb4dbf1ce19deba90a8df18.apk
./malware_app/c22ab450ae4ea678c4e8699ca0a83b4344e5ba3a.apk
./malware_app/9f83dab3edddf0cac9cc34844abaf1ccdbee4019.apk
./malware_app/d78f3bcf582b0db812ca3e94526455dc0c369fc5.apk
./malware_app/c4a8bb0a0f0a2ce4592656972b7043fc9b3a56d2.apk
./malware_app/b234dc00928261662f3df2ed46ed7eb44e359066.apk
./malware_app/90f568425cfcdea3fe19b3de93601eddc6bdc0e5.apk
./malware_app/890712525a30ac91ae42306a0a2a977d14b69d2e.apk
./malware_app/b349851d2a8ba476a0099c17714559f713aa2fdc.apk
./malware_app/8f7bf37face686ac456c21dc1dad132f077ce626.apk
./malware_app/b1675db6fbb245478bc14041cfcb48b0086071fb.apk
./malware_app/4f4ee687c683e889f204b1a0c86878f198380513.apk
```

2.3 Virus Total, Jadx를 사용한 앱 분석

Virus Total을 이용해 검사해본 결과, 우리가 분류했던 권한의 기준으로는 모든 앱에 정확히 일치하는 것이 아니라는 것을 발견했습니다. 예를 들어, 정상 앱으로 분류된 것임에도 불구하고 Virus Total로 열어봤을 때 수많은 악성 권한이 담겨있는 경우도 검출되었습니다.

또한, Jadx 도구를 사용해 역공학을 이용하여 정확히 어떤 메소드에서 문제를 일으키고있는지 역시 추적을 해보았고 그것을 토대로 행동패턴을 분석할 수 있었습니다.

2.3.1 악성 앱 분석결과 1

어플리케이션	53dc08f08005f374a957afa44607ab52f205b684.apk
권한	android.permission.INTERNET android.permission.READ_CONTACTS android.permission.READ_PHONE_STATE android.permission.WRITE_EXTERNAL_STORAGE
	android.permission.MODIFY_PHONE_STATE android.permission.WRITE_APN_SETTINGS android.permission.ACCESS_NETWORK_STATE android.permission.ACCESS_WIFI_STATE android.permission.RECEIVE_BOOT_COMPLETED
AndroidManifest.xml 악성 의심 클래스	<action android:name="android.service.wallpaper.WallpaperService" /> <action android:name="android.intent.action.PHONE_STATE" />

문제코드	TelephonyManager mTelephonyMgr
	= (TelephonyManager) getSystemService("phone");
	this.imei = mTelephonyMgr.getDeviceId();
	this.imsi = mTelephonyMgr.getSubscriberId();

행동	배경화면을 바꾸는 어플로 위장해 디바이스의 정보를 빼냄

2.3.2 악성 앱 분석결과 2

어플리케이션	b1675db6fbb245478bc14041cfcb48b0086071fb.apk
권한	android.permission.INTERNET android.permission.READ_PHONE_STATE
	android.permission.GET_ACCOUNTS android.permission.RECEIVE_BOOT_COMPLETED
AndroidManifest.xml 악성 의심 클래스	<service android:label ="@string/service_name"android:name="com.and.snd.MosquitoSoundService" android:enabled="true">
문제코드	TelephonyManager telephonyManager = (TelephonyManager) context.getSystemService("phone"); this.deviceId = telephonyManager.getDeviceId(); String msisdn = telephonyManager.getLine1Number();
행동	모기소리를 내는 어플로 위장하여 어플 실행 시 디바이스의 정보를 획득

2.3.3 악성 앱 분석결과 3

어플리케이션	a8634a3392690a13146fbfe286e117b3e34bf96e.apk
권한	android.permission.INTERNET android.permission.READ_PHONE_STATE
	android.permission.GET_ACCOUNTS android.permission.RECEIVE_BOOT_COMPLETED
AndroidManifest.xml 악성 의심 클래스	<service android:label="@string/service_name" android:name="com.and.snd.WhoopeeSoundService" android:enabled="true">

문제코드	this.accounts = AccountManager.get(getApplicationContext()).getAccounts();
	this.country = telephonyManager.getNetworkCountryIso().toUpperCase();
	this.carrier = telephonyManager.getNetworkOperatorName();
	AccountManager.get(getApplicationContext()).getAccounts();

행동	디바이스의 네크워크 정보나 계정정보 획득.

2.4 악성 앱 탐지 프로그램의 구현

2.4.1 소스코드

기존의 feature_extractor 코드를 이용하여 권한정보 탐색 시에 악성 권한이 검출된다면 악성 앱으로 나타나도록 코드를 추가하였습니다. Smail.py에서 Androguard를 통해 권한정보를 모두 권한만 담겨있는 가공된 문자열 배열로 반환해 주기 때문에 정규식을 사용하지 않고 단순 문자열 비교를 통해 확인할 수 있었습니다. 해당코드에서 24번 라인에서 확인할 수 있는 malwarePermissionList에는 위에서 얻어낸 악성 앱의 특징적인 권한정보가 담겨있습니다.

```
1  class DetectionManager(object):
2    def __init__(self,args,fname,dvm,vma,temp_d):
3      self.args = args
4      self.filename = fname
5      self.DetectionResult = 0
6      self.Permission = []
7      self.APICheck = []
8
9      if args.All:
10       args.Smali = True
11       args.APICheck = True
12
13     if args.Smali:
14       self.Permission = getSmali(dvm, vma, self.filename)
15     if args.APICheck:
16       self.APICheck = getAPICheck(dvm, vma)
17
18   def doSaveAll(self,rname):
19     f = open(rname,'ab')
20     f.write(str(self.filename)+',')
21     self.Permission_final = set(self.Permission)
22 #추가한 영역
23     detectedPermissionList = []
24     for pl in malwarePermissionList:
25       for p in self.Permission_final :
26         if p == pl:
27           detectedPermissionList.append(p)
28     if len(detectedPermissionList) == 0 :
29       print 'test('+rname+') : 정상앱입니다!'
30       f.write('This is benign App!,')
31       else :
32       print 'test('+rname+') : 악성앱입니다!'
33       f.write('This is malware App!,')
34     for pm in detectedPermissionList:
35       f.write(str(pm)+',')
36     f.write(str(self.APICheck)+',')
37     f.write(str(self.DetectionResult)+',')
38     for pm in self.Permission_final:
39       f.write(str(pm)+',')
40     f.close()
```

2.4.2 실행결과

	A	B	C
1	./malware_app/047043d752265fce625d22113e0b70dfb5e88477.apk	This is malware App!	android.permission.MOUNT_UNMOUNT_FILESYSTEMS
2	./malware_app/54dacf873755f45d77b2e64e56bd070718f5aaa2.apk	This is malware App!	android.permission.INSTALL_PACKAGES
3	./malware_app/8784ee14bd5f4e1ef31073cc42bde7fa6671da43.apk	This is malware App!	android.permission.RESTART_PACKAGES
4	./malware_app/bc2dedad0507a916604f86167a9fa306939e2080.apk	This is malware App!	android.permission.CHANGE_NETWORK_STATE
5	./malware_app/40156a176bb4554853f767bb6647fd0ac1925eac.apk	This is malware App!	android.permission.RESTART_PACKAGES
6	./malware_app/03f9fc8769422f66c59922319bffad46d0ceea94.apk	This is malware App!	android.permission.CHANGE_NETWORK_STATE
7	./malware_app/ecd91723a885c36d6c03ef4cbeb2b8b0c8177b6f.apk	This is malware App!	android.permission.INSTALL_PACKAGES
8	./malware_app/f4fc04c1e1566c80875160236641cd5b84f7da57.apk	This is malware App!	android.permission.RESTART_PACKAGES
9	./malware_app/edad14044d5e0b055ead541700052f3d4a25cd41.apk	This is malware App!	android.permission.CAMERA
10	./malware_app/bd7b1c30019219f8c5516976df9429f8ab7e59e8.apk	This is malware App!	android.permission.MOUNT_UNMOUNT_FILESYSTEMS
11	./malware_app/eb32bc6a768db927af6555c000498c07f796b419.apk	This is malware App!	android.permission.CHANGE_NETWORK_STATE
12	./malware_app/dc46fefc4cb3d0fc0fcaa4fba76e5d5c7538da2f.apk	This is malware App!	android.permission.RESTART_PACKAGES
13	./malware_app/fcd71343eed2c5471870af8153eb4176b6893b73.apk	This is malware App!	android.permission.RESTART_PACKAGES
14	./malware_app/806aaf7772889ae61d64dd8c40e674fc3db497b7.apk	This is benign App!	
15	./malware_app/e8e171e053c83e628041c7d57059e040da13fa0e.apk	This is malware App!	com.lge.launcher.permission.READ_SETTINGS
16	./malware_app/73ddc408b518826064878dfc0064c4cd4fe512c0.apk	This is benign App!	
17	./malware_app/aca8b04659e909d9bb2e35c6baba768d8fdad3d0.apk	This is malware App!	android.permission.INSTALL_PACKAGES
18	./malware_app/132f70936801d48da0677dedb09ef526ff1403ee.apk	This is malware App!	android.permission.CLEAR_APP_CACHE
19	./malware_app/8b22ff8358906925bd070903aa0bf4d989af7532.apk	This is malware App!	com.lge.launcher.permission.READ_SETTINGS
20	./malware_app/6a0bfabcc1cce2a5424313b34ca967fbc8f98bea.apk	This is malware App!	android.permission.WRITE_APN_SETTINGS
21	./malware_app/ffe62967b75aab56710110b26baa69acd47a81dd.apk	This is malware App!	android.permission.WRITE_APN_SETTINGS
22	./malware_app/15799032055df9f93440b61547b37e01a7234a0c.apk	This is benign App!	
23	./malware_app/202252e3767456e95b27d0476ae29bcc11253c1e.apk	This is malware App!	com.lge.launcher.permission.READ_SETTINGS
24	./malware_app/0db53aefbd60325df198e9ebb190f45c61ac8923.apk	This is benign App!	
25	./malware_app/4626899e1c247b9b6ad6b9ebb2ef67db41591563.apk	This is malware App!	android.permission.RESTART_PACKAGES
26	./malware_app/d64419d569a336f05d0eb799615097870934c42a.apk	This is benign App!	
27	./malware_app/113d68bbaa7ab1f0abd9f9a5b772f03ea6721e14.apk	This is malware App!	android.permission.CHANGE_NETWORK_STATE
28	./malware_app/005d5f6e94321de473d62706a94fbecf67c9f5f3.apk	This is benign App!	
29	./malware_app/63e642f0d859e096342321c9e03baca7cd1210fa.apk	This is malware App!	android.permission.PERMISSION_NAME
30	./malware_app/7ded7bb7041acce78e85e811c0ac048338bdc2d9.apk	This is malware App!	android.permission.RESTART_PACKAGES
31	./malware_app/de538dc6be84effaafa8512c1ae361e8613ffeee.apk	This is malware App!	android.permission.MOUNT_UNMOUNT_FILESYSTEMS
32	./malware_app/e9c841757db6816f978deb573fd8e8ed0fe6be3b.apk	This is malware App!	android.permission.CHANGE_NETWORK_STATE
33	./malware_app/0c059ad62b9dbccf8943fe4697f2a6b0cb917548.apk	This is benign App!	
34	./malware_app/2e998614b17adbafeb55b5fb9820f63aec5ce8b4.apk	This is malware App!	android.permission.RESTART_PACKAGES
35	./malware_app/c9305cc8e1d2908708a49e8d9b932c2fdb952106.apk	This is malware App!	android.permission.INSTALL_PACKAGES
36	./malware_app/edbbc205e7033174919ba22b01617fd731960da2.apk	This is benign App!	
37	./malware_app/7f11794529652cf3e785e1c51b9ccbf1f00bce4a.apk	This is benign App!	
38	./malware_app/d2fdaf2c293adb2977c8d3c7aa7152553021f949.apk	This is benign App!	
39	./malware_app/1c0a6b1c5d24cbba9b11020231fffc0840dd7e10.apk	This is benign App!	
40	./malware_app/8fc9a38da971f3524ceb40816b9c31a69c790e98.apk	This is malware App!	android.permission.RESTART_PACKAGES
41	./malware_app/9797079492eb16667cadc4f1b17ca9b0fbe4a967.apk	This is malware App!	com.lge.launcher.permission.READ_SETTINGS
42	./malware_app/aa2218213aa8fbdb9b502e2ce03bbaf8cb494b30.apk	This is malware App!	com.lge.launcher.permission.READ_SETTINGS
43	./malware_app/9d811ebf06dfb749209490b70c302f88c3a811dc.apk	This is malware App!	android.permission.RESTART_PACKAGES
44	./malware_app/b14de36c493240f98cc610982b4ff72d9939907a.apk	This is malware App!	android.permission.CHANGE_NETWORK_STATE
45	./malware_app/426f501c28e6c24d755d71c07561ebf7f1aba5fc.apk	This is malware App!	android.permission.MOUNT_UNMOUNT_FILESYSTEMS

결과 csv 파일에 benign application 인지 malware application 인지 구분해주고, malware application 이라면 문제가 되는 permission 을 작성해주도록 바꾸었습니다. 2.2.3 에서 찾았던 악성 앱이지만 정상 권한만을 갖는 30 개 앱은 아쉽게도 권한만으로는 걸러낼 수 없는 점을 확인할 수 있습니다.

3. 과제를 수행함에 있어서 어려웠던 점, 건의사항

- 처음 악성 앱과 정상 앱 100개씩을 받은 후, 이를 처음 압축을 해제하는 과정에서 오류가 검출
- 조교님으로부터 수정된 소스코드를 받기 전에 실험을 하였을 때, 퍼미션들이 계속적으로 누적되는 현상이 발견하여 제대로 된 결과가 검출되지 않았음.
- 안드로이드 스튜디오를 설치하는 과정에서 경로이름이 한글로 되어있어 아스키문자열로 인식을 못하여 경로이름을 새로이 재지정했어야 하는 어려움
- feature_extractor를 이용하여 얻은 permission중 어떤 퍼미션이 정확히 어떠한 악성권한을 가지는지 구분하는데 어려움을 겪어서, 이에 대한 정확한 분류를 검색을 통해 알아보았음.
- Virus Total을 이용해 검사해본 결과, 우리가 분류했던 permission의 기준으로는 모두가 정확히 일치하는 것은 아니라는 것을 발견
- Android application에 대하여 배우지 않은 상태에서 낯선 코드들을 분석하고 이해하는 부분이 힘들었습니다.